Introduction:

In the world of Customer Relationship Management (CRM), establishing a secure and reliable connection between CRM systems and external applications is crucial for seamless data integration and streamlined workflows. However, connecting a CRM system, such as Microsoft Dynamics 365, with a console application can often present challenges. In this blog post, we will explore the common issues encountered when connecting a CRM system using a console app and discuss how leveraging Azure's App Registration can help overcome these challenges.

Problem Statement:

Additional Point: Deprecation of Other Authentication Methods - Embracing OAuth.

In recent updates, Microsoft has deprecated several legacy authentication methods and encourages developers to adopt OAuth 2.0 as the standard authentication mechanism for connecting applications to CRM systems, including Microsoft Dynamics 365. As a result, it is essential to note that OAuth 2.0 is the recommended and supported approach for authenticating and authorizing console applications when connecting to CRM systems.

Connecting a CRM system with a console application poses various challenges, including:

Authentication: Console applications typically lack user interfaces, making it challenging to establish secure authentication with CRM systems that require user credentials. How can we authenticate and securely connect a console app to a CRM system?

Authorization: Console applications may require specific permissions or roles to access and interact with CRM data. How can we ensure the console app has the necessary authorization to perform desired operations on the CRM system?

Endpoint Configuration: CRM systems often have complex endpoint configurations, including authentication endpoints, resource endpoints, and API versions. How can we properly configure the endpoints within the console application to establish a successful connection with the CRM system?

Secure Storage of Secrets: Console applications often require sensitive information, such as client IDs, client secrets, and tenant IDs, to establish the connection. How can we securely store and retrieve these secrets within the console app to prevent unauthorized access?

Solution:

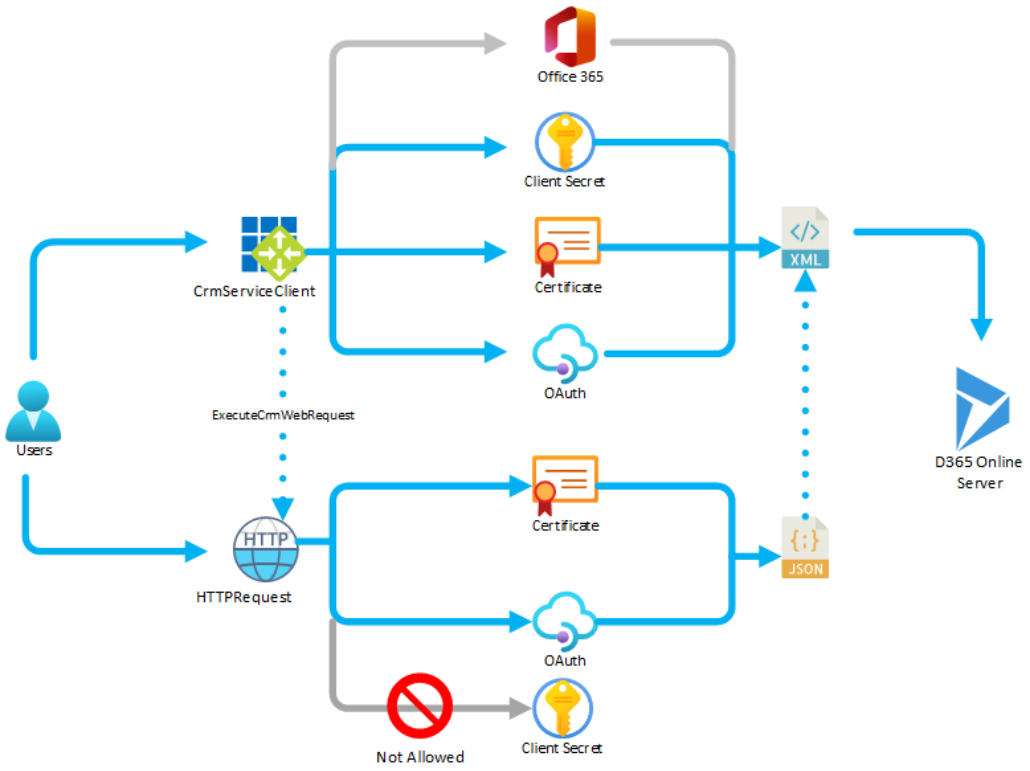
pre-requisites:

Microsoft Dynamics 365 Sales Trial - This will provide you with the necessary environment to connect your console application to the CRM system.

Azure Subscription - If you don't have an Azure subscription, you can create a free account or use an existing one.

Visual Studio or Equivalent IDE –

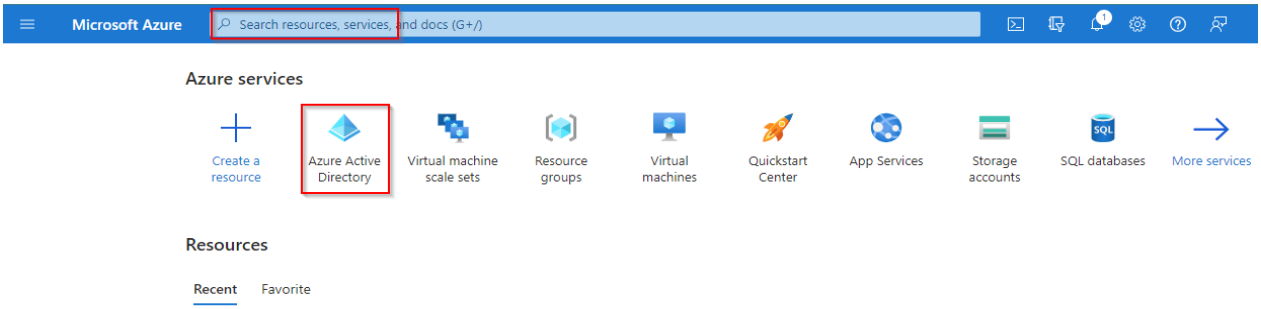
Architecture -



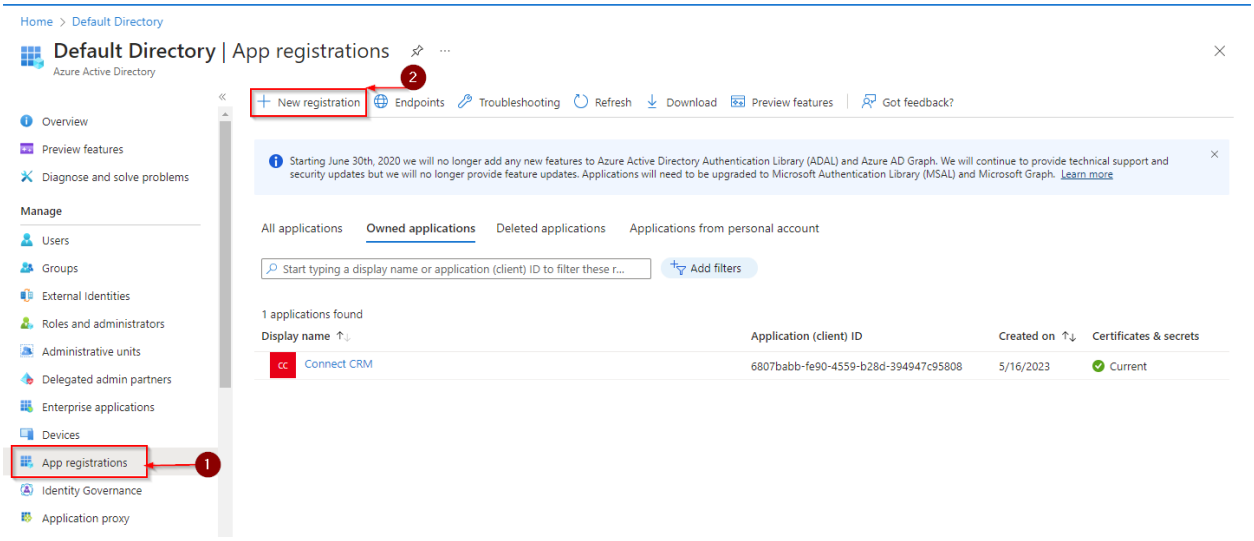
Step 1: Configure App registration in Azure Active Directory.

Sign in to the Azure portal: Go to the Azure portal (portal.azure.com) and sign in with your Azure account credentials.

Navigate to Azure Active Directory (**Azure AD**): In the Azure portal, search for and select "Azure Active Directory" from the list of services.

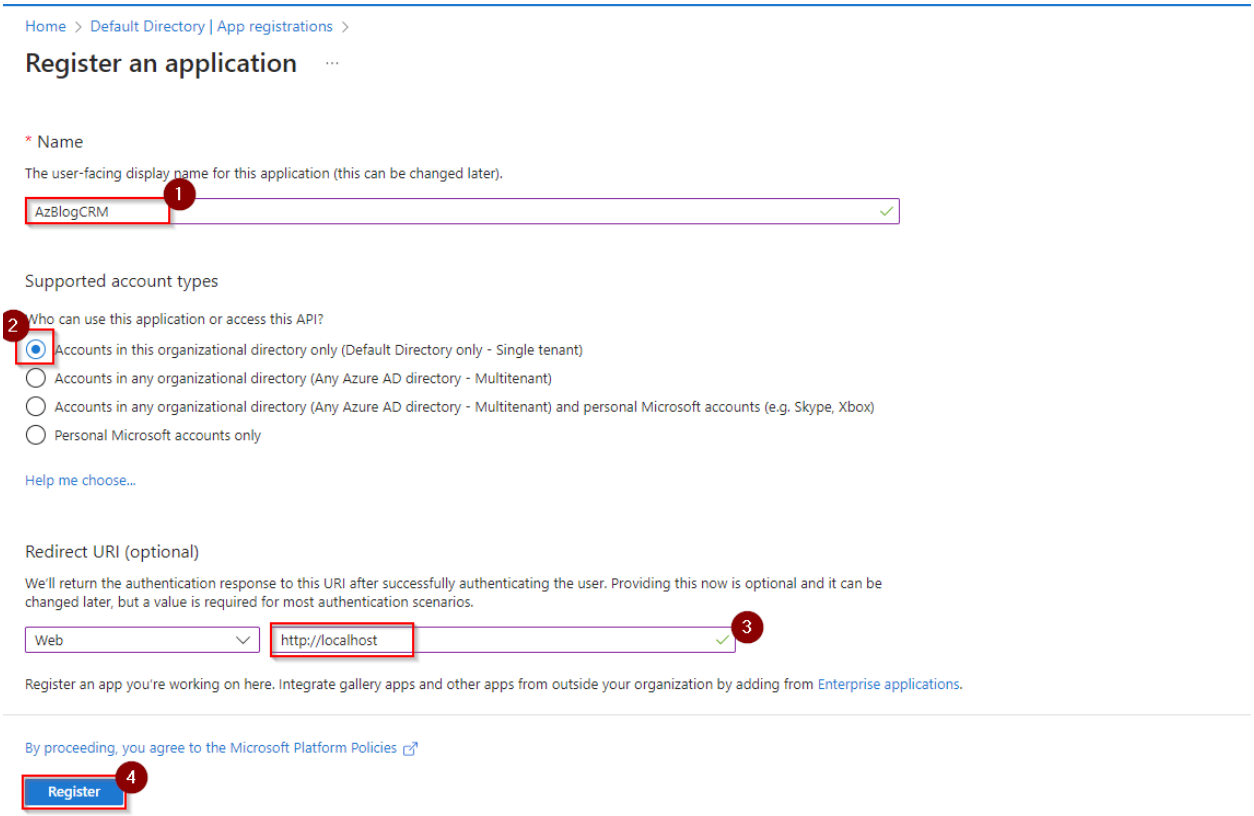


Access App Registrations: In the Azure AD pane, select "App registrations" from the left-hand side menu. Click on the "New registration" button to start creating a new App Registration.

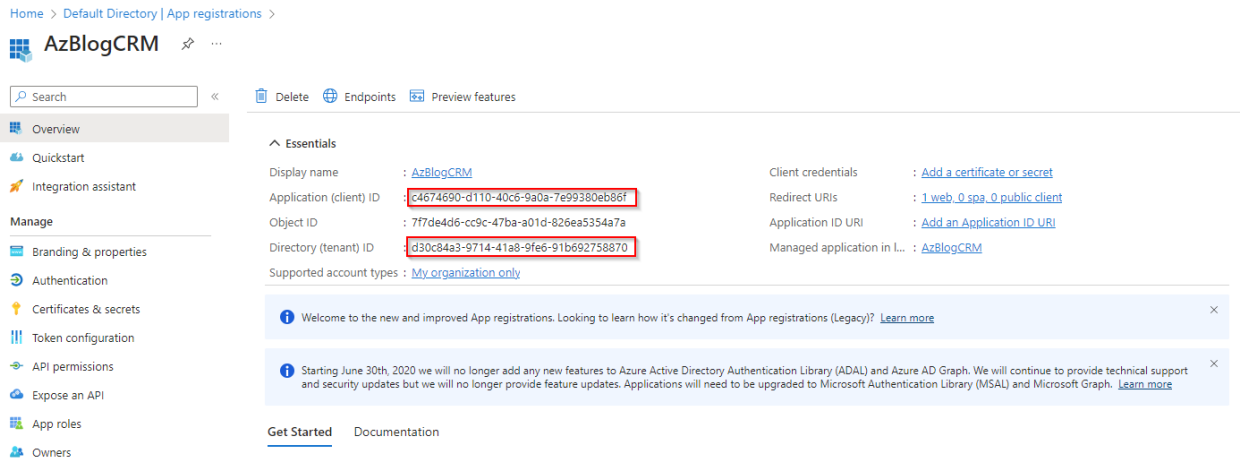


Provide registration details: In the "Register an application" form, provide the following details:

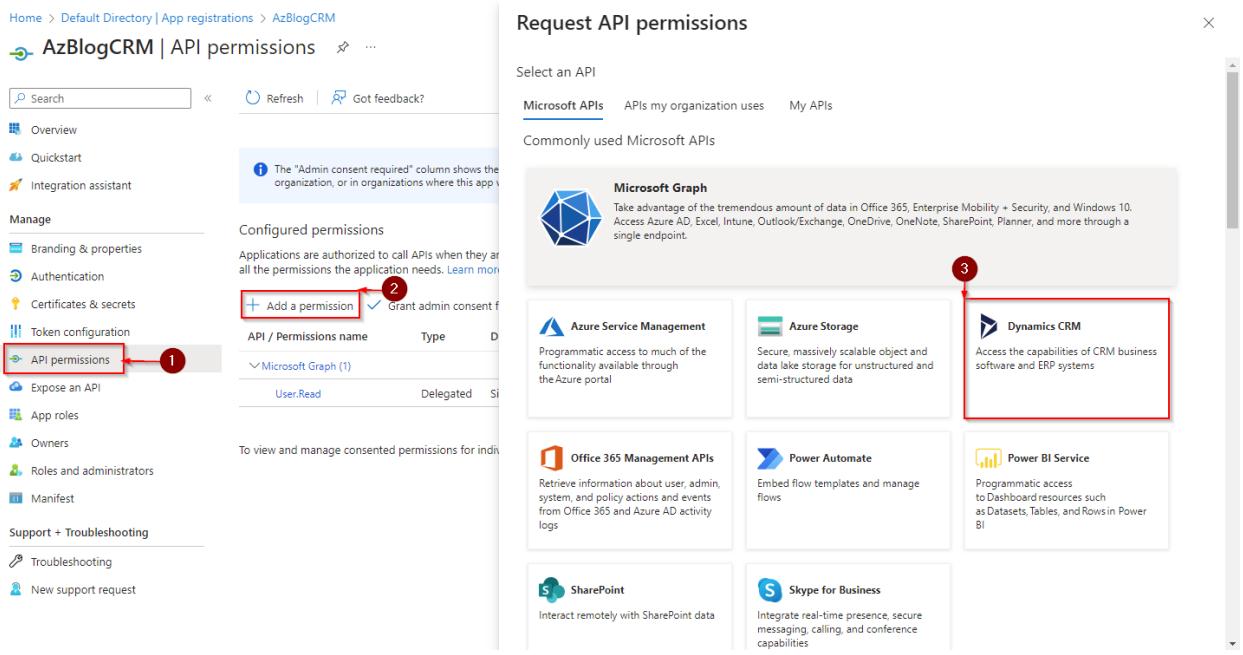
Enter an application name - Give your application a meaningful name.
Supported account types - Choose the appropriate account types for your scenario (e.g., single tenant, multi-tenant, personal Microsoft accounts).
Redirect URI (optional) - Specify the redirect URI if your application requires one for authentication.
Register the application: Click on the "Register" button to create the App Registration.



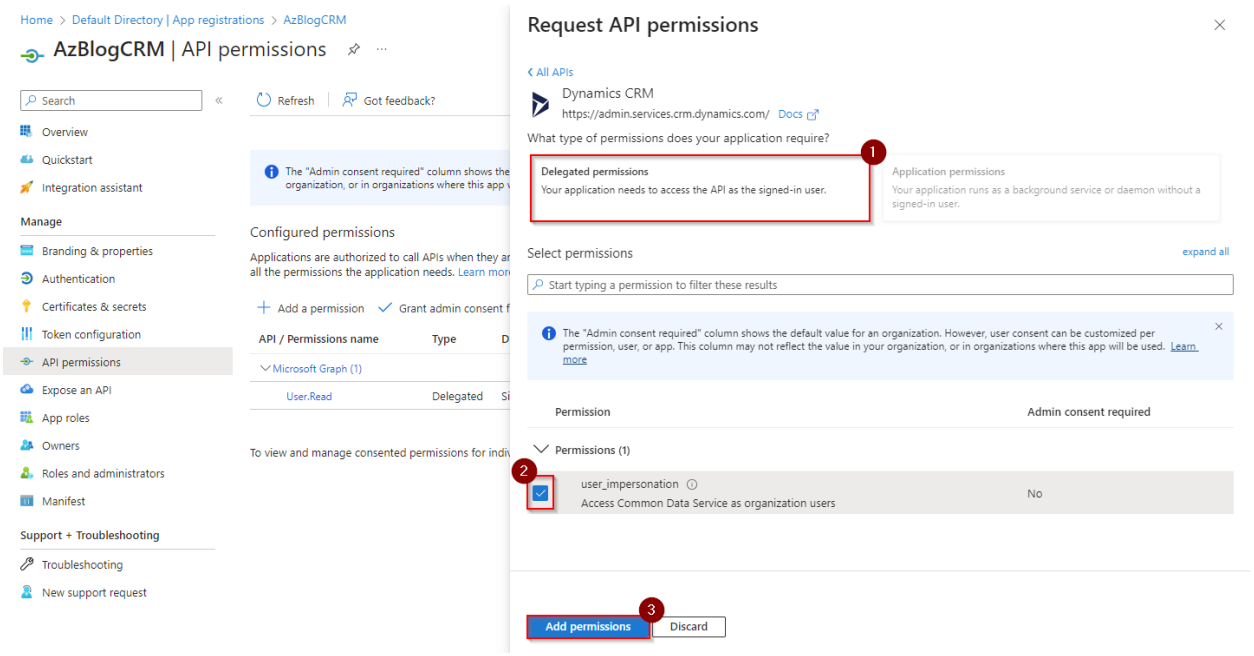
Obtain the client ID: After the registration is complete, you will be redirected to the Overview page for your App Registration. Note down the "Application (client) ID" value, as you will need it for authentication and authorization.



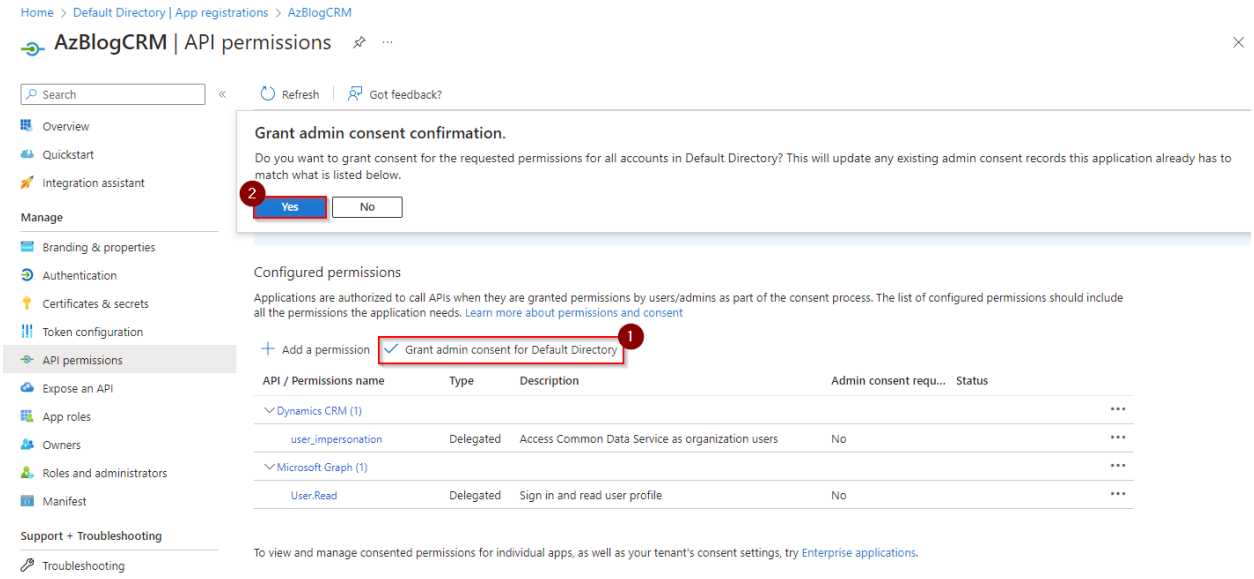
Configure API permissions: Next we have to provide API permission to the APP. on same App page go to **API Permissions** option Click on **Add Permission** and in the list Select **Dynamics CRM**.



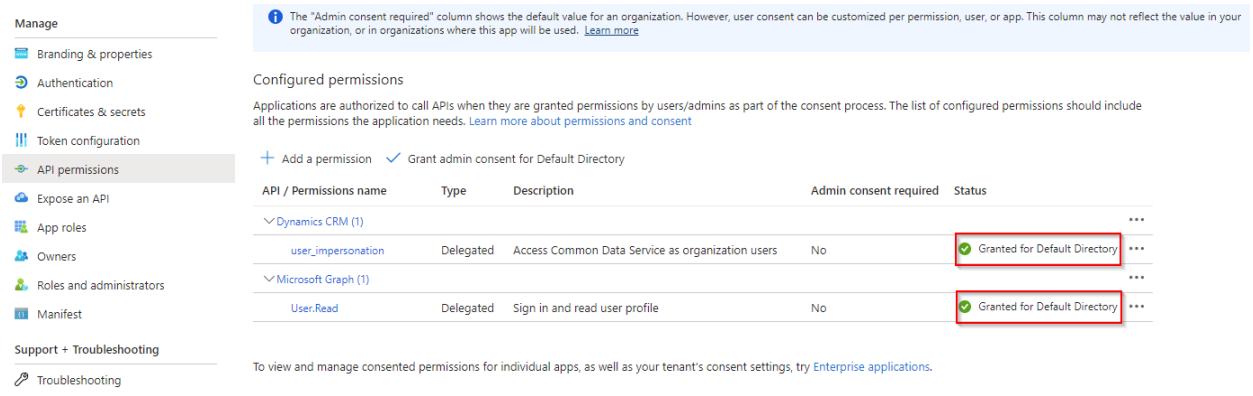
Then Select **Delegated Permission**. Select the **User Impersonation** permission. Then click **Add Permission** button.



Next, we have to Grant Admin Consent for the permission. Click the **Grant** admin consent and click **Yes**. This the necessary for the app to function -

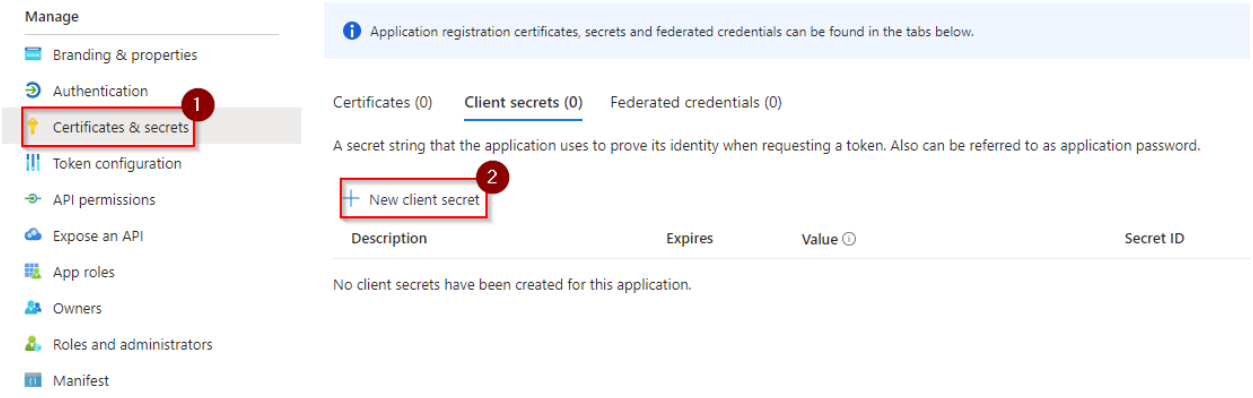


Now, the status will display as Granted. Now we are ready to process next step.

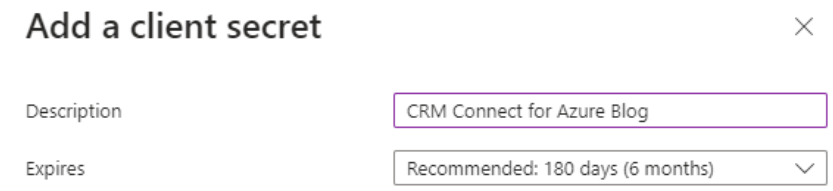


Generate and securely store a client secret: If your console application requires a client secret, select "Certificates & secrets" from the left-hand side menu under your App Registration.

Click on the "New client secret" button.

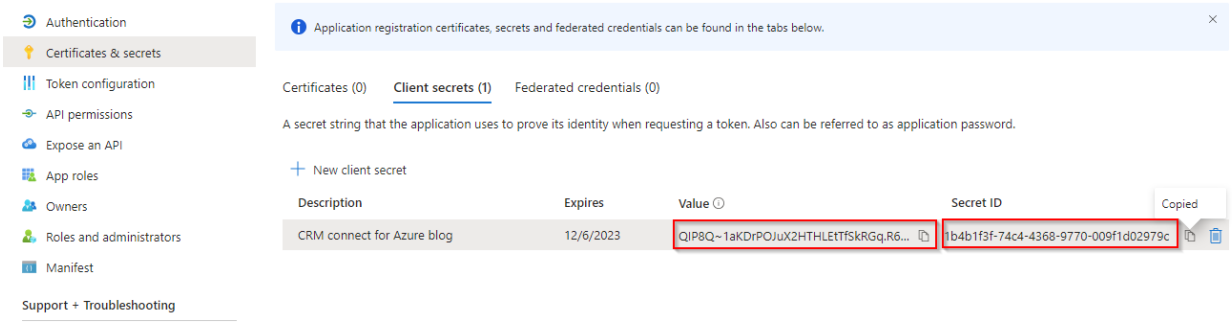


Enter a description for the client secret and select an expiration period. Click on the "Add" button to generate the client secret. Note down the generated value as it will be needed for your console application's authentication.



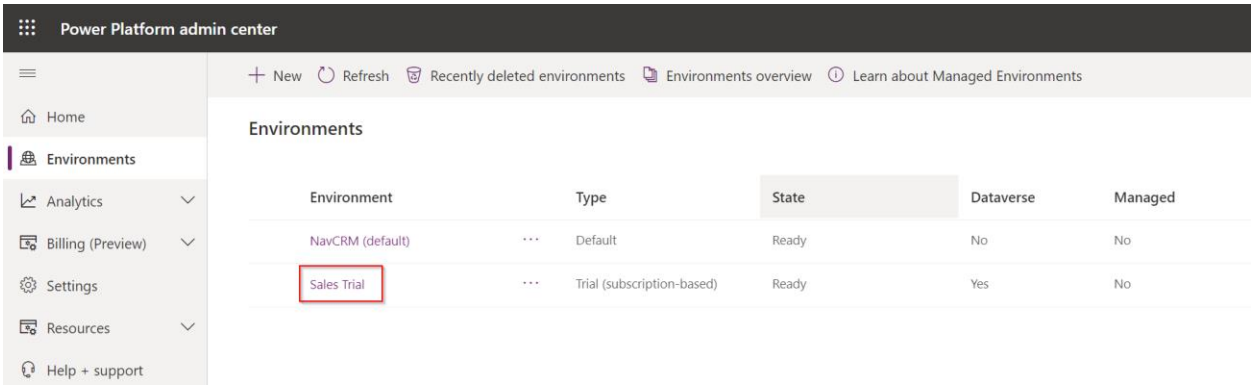
Tip - Now quickly copy the Secret ID and value. Because if you do not copy after sometime you will not get the value.

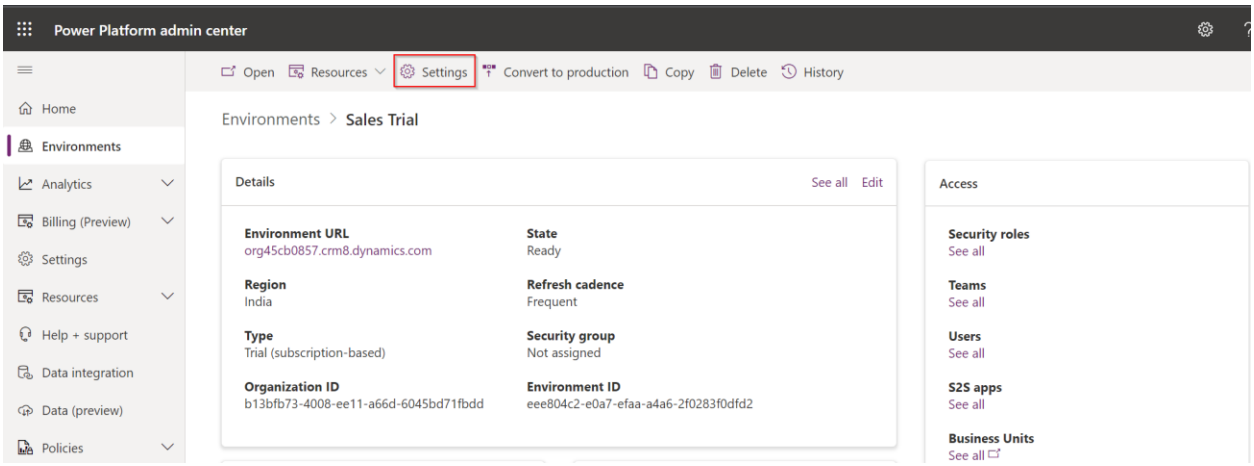
This Secret Value is required in our code.



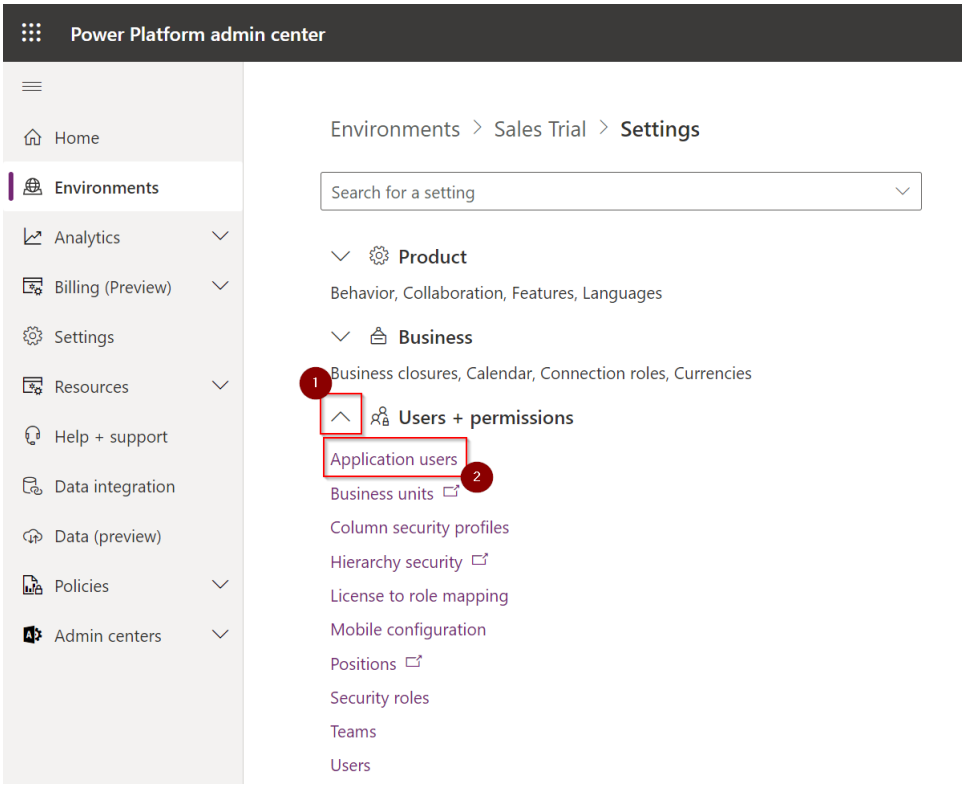
Step 2: Configure Application user in Dynamics 365 Admin Centre.

Go to **'admin.powerplatform.microsoft.com'** Click on the desired environment then click on settings. (In this way we don't need to add Application ID it will be automatically added)-

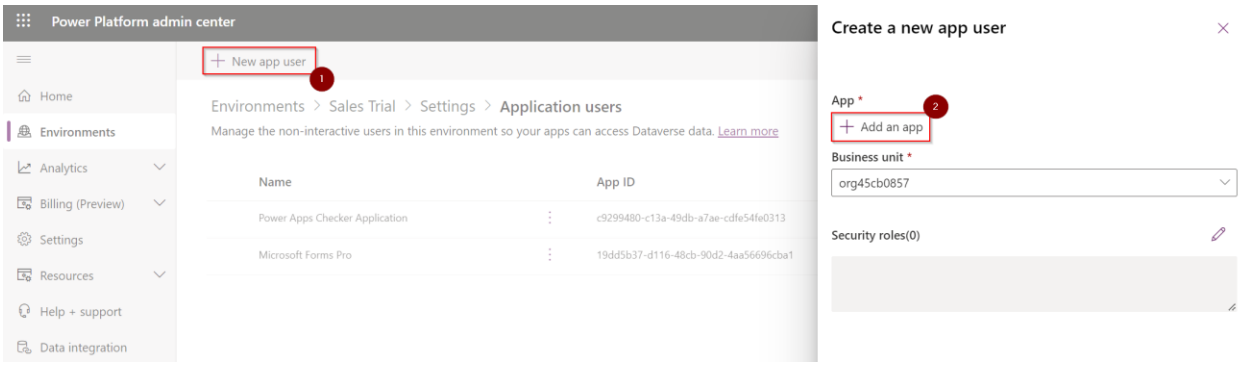




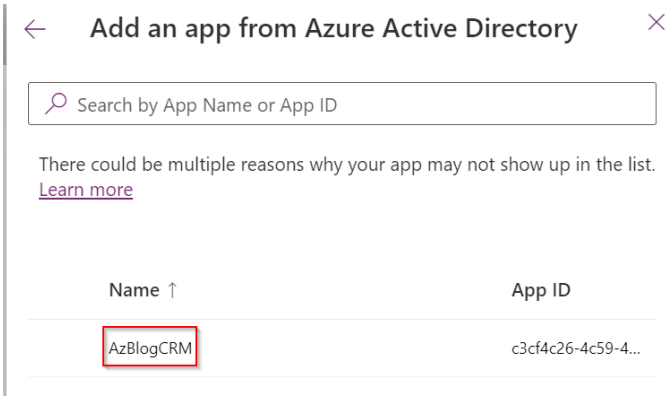
Now, click on **'User + permissions'** and click **Application user** -



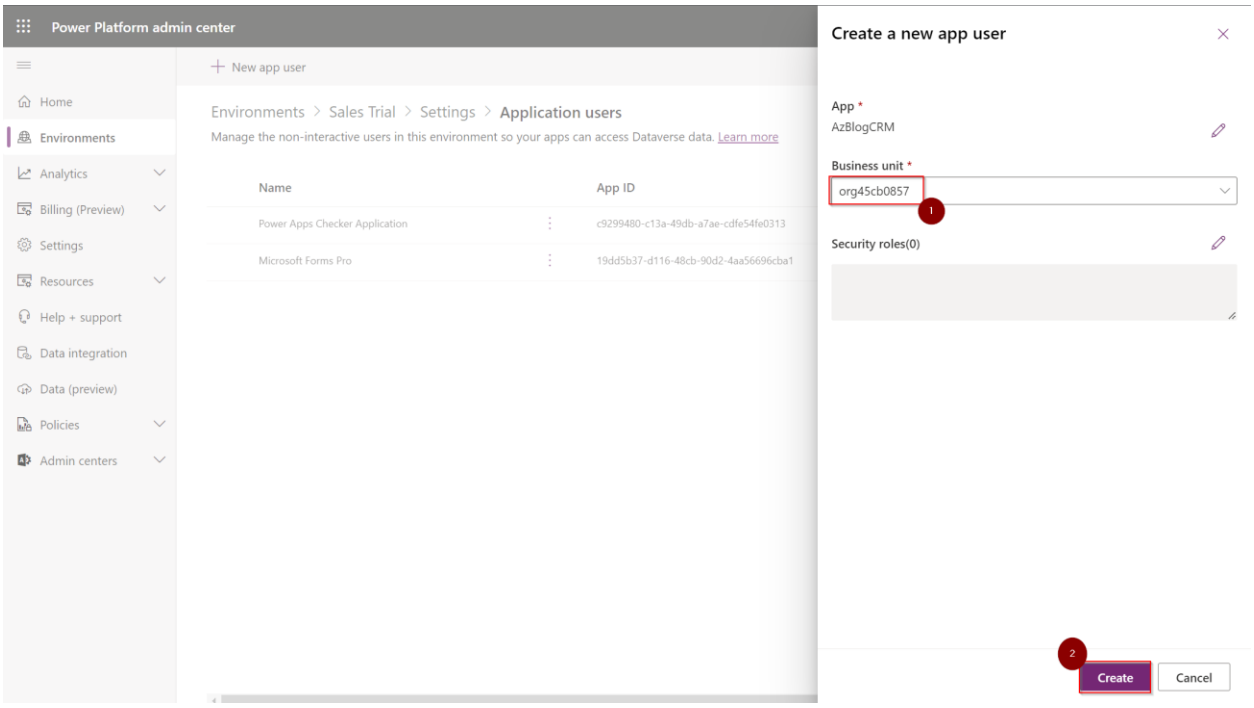
Click on **+New app** user and then click on **+Add app** –



This view will open. The app registered on Azure earlier will be visible here click on it –

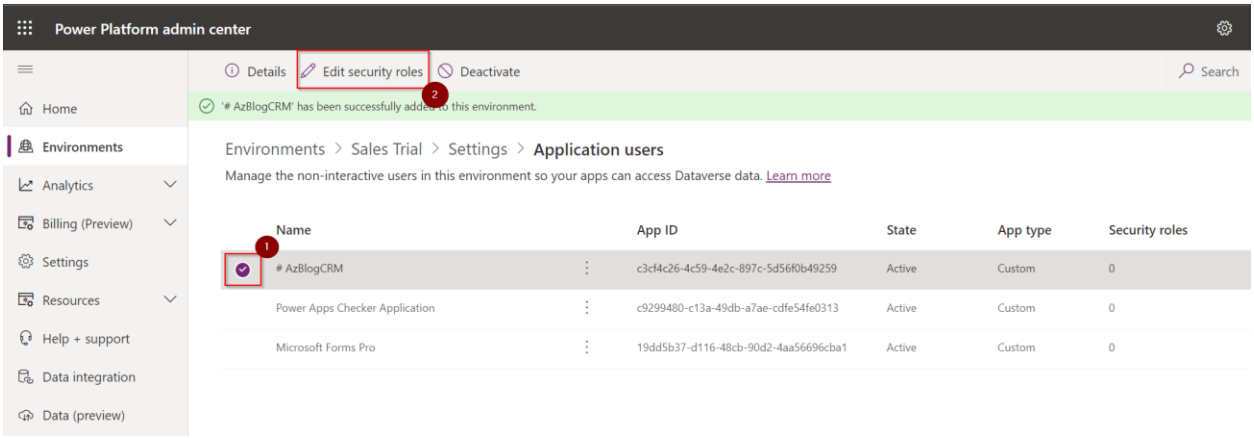


Check the business unit same as the environment you working with then click **Create** –



Now, edit the security role to assign a role to the app to use the desired features –

Select the App then select Edit security roles –

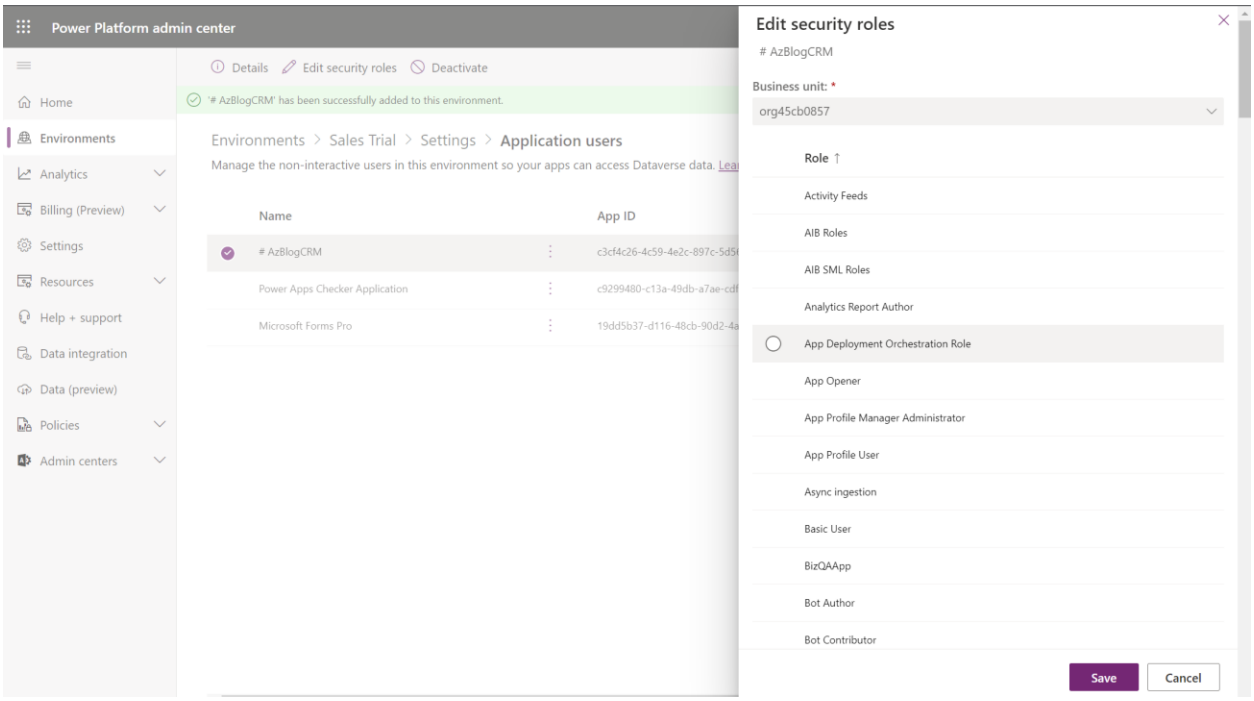


Now, choose the specific roles or permissions you want to assign to the user.

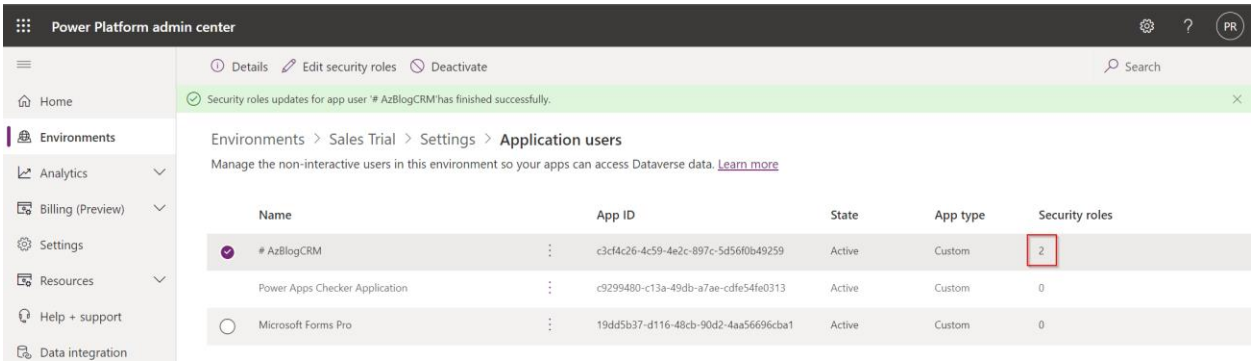
This step allows you to define the level of access and capabilities the user will have when interacting with the Dynamics CRM system.

Select the appropriate roles based on your requirements and the operations the console application needs to perform on the CRM data.

Select the roles you want to assign to the user –



In my case I have given **System customizer, system administration** since I have to read write data in excel, you can give any according to your use case -



We have successfully registered the app now we are ready to connect CRM using Console Application.

Step 3: Create a console Application in C# and use Client ID and Client Secret for CRUD operation.

Create console application and add required Microsoft assemblies using NuGet Package Manager.

Once added all Microsoft XRM assemblies modify the **Program.cs** class file to add below code.

You will need **"organizationUri"** of your environment and **"clientId" "clientSecret"** you copied and saved earlier from azure.

```
using System;
using System.Configuration;
using System.Security.Cryptography;
using System.Text;
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;
using Microsoft.Xrm.Tooling.Connector;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Console App started");

        IOrganizationService orgService;

        orgService = GetOrganizationServiceClientSecret(
            "c77c4ed4-d0a4-4c1f-b73f-1328ec49e21e",
            "W2R8Q~cqlcQkXdBhdxA_8Xzt7Fq0OnhmkwiJ_ch~",
            "https://org45cb0857.crm8.dynamics.com/");

        if (orgService != null)
        {
            Console.WriteLine("Connection Successful!...\n");

            Entity account = new Entity("account");
            account["name"] = "Mohit Account";
            var createacc = orgService.Create(account);

            Console.ReadKey();
        }
        else
        {
            Console.WriteLine("Failed to Establish Connection!!! \n");
            Console.ReadKey();
        }
    }

    public static IOrganizationService GetOrganizationServiceClientSecret(string
clientId, string clientSecret, string organizationUri)
    {
        try
        {
            var conn = new
CrmServiceClient($@"AuthType=ClientSecret;url={organizationUri};ClientId={clientId};Clien
tSecret={clientSecret}");

            return conn.OrganizationWebProxyClient != null ?
conn.OrganizationWebProxyClient : (IOrganizationService)conn.OrganizationServiceProxy;
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error while connecting to CRM " + ex.Message);
            Console.ReadKey();
            return null;
        }
    }
}
```

Challenges Faced:

Authentication and Authorization: One of the main challenges when connecting a CRM system with a console application is implementing secure and reliable authentication and authorization. Managing user credentials, ensuring secure access, and handling tokens can be complex and prone to errors.

Endpoint Configuration: Configuring the correct endpoints for the CRM system within the console application can be challenging. Incorrect endpoint configuration can result in connection failures or data retrieval issues.

Secure Storage of Secrets: Storing sensitive information, such as client IDs and client secrets, within the console application securely is crucial. Without proper measures, storing secrets within the application code can expose them to potential security risks.

Business Benefits:

Streamlined Data Integration: Connecting a console application to a CRM system enables seamless data integration. This allows businesses to consolidate data from various sources, providing a unified view of customer information, enhancing decision-making, and improving operational efficiency.

Automation of CRM Tasks: By connecting a console application to a CRM system, businesses can automate repetitive or time-consuming CRM tasks. This reduces manual effort, improves productivity, and ensures consistency and accuracy in CRM-related processes.

Customized Workflows and Business Logic: A connected console application allows businesses to implement custom workflows and business logic tailored to their specific needs. This flexibility enables the automation of unique business processes, ensuring optimal utilization of the CRM system.

Enhanced Data Synchronization: Console applications can facilitate seamless data synchronization between the CRM system and external databases or applications. This ensures that data remains consistent across multiple systems, providing a holistic view of customer information and supporting data-driven decision-making.

Improved Customer Experience: Connecting a console application with a CRM system enables businesses to provide a more personalized and efficient customer experience. Access to real-time CRM data empowers customer service representatives to better understand customer needs, resolve issues promptly, and deliver personalized interactions.

References –

(Reference: https://docs.microsoft.com/en-us/azure/active-directory/)
(Reference: https://docs.microsoft.com/en-us/dynamics365/)